Transitioning Your Organization

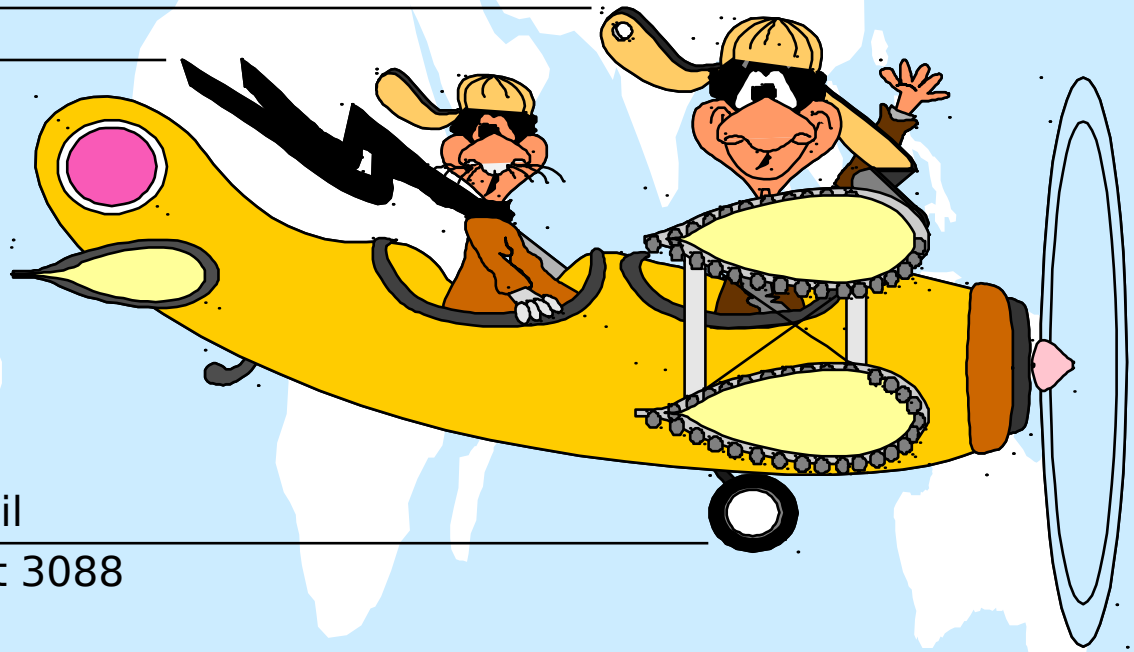
to an

Object-Oriented Methodology

#*

Speakers

David A. Cook
USAF STSC/Draper Labs
cookd@software.hill.af.mil
Phone (801) 775-5555, ext 3086

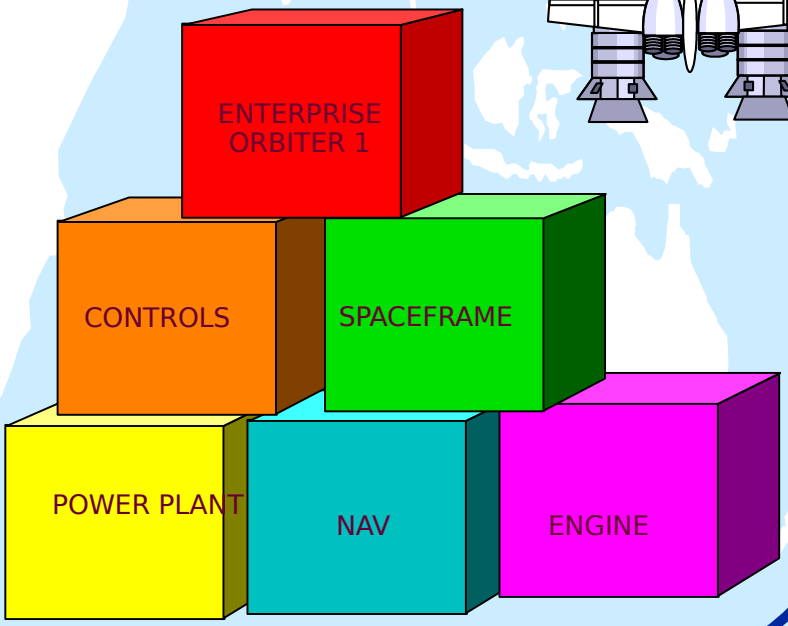


Leslie (Les) Dupaix
USAF STSC
dupaixl@software.hill.af.mil
Phone (801) 775-5555, ext 3088

#*

Why transition to OO?

- OO methodologies attempt to build complex systems using “classes” and “objects” as the building blocks
  - Attempts to model the real world via abstraction
  - Evolutionary, not revolutionary
- Experience has shown that this is the superior way to manage and decompose an inherently complex system. Functional methodologies require you to understand the entire system before you can modularize it!



The OO Paradigm

Macro View

- Identify the core requirements for the software (conceptualization)
- Develop a model of the system's desired behavior (analysis)
- Create an architecture for the implementation (design)
- Evolve the implementation through successive refinements (evolution)
- Manage postdelivery evolution (maintenance)

#*

The OO Paradigm

Micro View

- **Identify the classes at a given level of abstraction (OOD issue)**
- **Identify the objects at a given level of abstraction (OOD issue)**
- **Identify the relationships among these classes and objects (OOD issue)**
- **Specify the interface and then code the implementation of these classes and objects (OOP issue).**

#*

Benefits of OO

Reuse

Well-designed classes can be extended via inheritance. Previous code can be reused without modification to users of the original code.

Allowing lots of subclasses allows special cases to be truly special without modifications to normal classes.

Integration

OO supports incremental development.

Additions to the system are easier.

It is easy to develop the parent class first, and then concentrate on subordinate classes.

#*

Benefits of OO (cont.)

Testing

Easier to test, as specific behaviors and classes can be tested individually.

Specific test cases can be designed to test a class, and then this behavior does not need to be retested for subordinate classes

Maintenance

Additions to the system can be accomplished without modifying existing code.

Instead of modifications to existing classes, new classes can be constructed

OOSE - Myths and Facts

- Myth - OO can be inserted into programming projects currently under development
- Myth - OO will provide reusable code, thus lowering my overall system cost
- Myth - OO will work with existing methodologies and syste

OOSE - Myths and Facts

- Fact - OO must be part of the overall lifecycle to recognize savings

- Fact - Reuse must be designed into code. It cost more to make reusable code.

- Fact - OO requires supporting CASE tools and extensive training to realize its full benefit

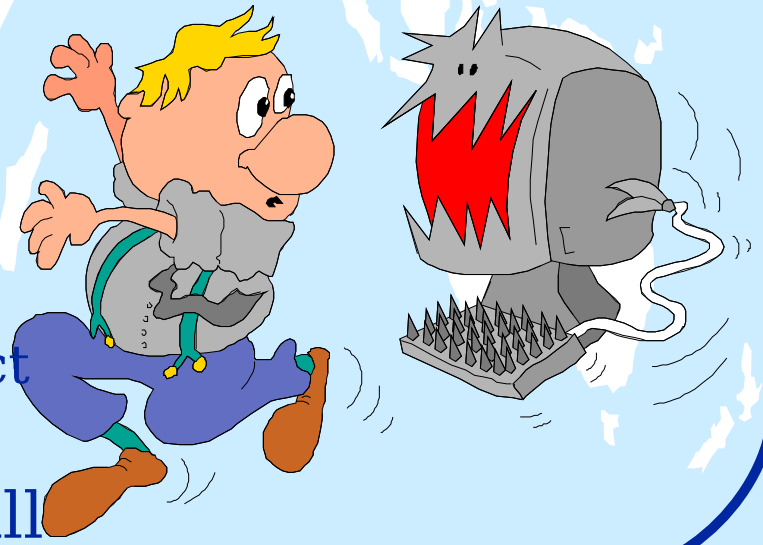
THE BOTTOM LINE

The insertion of OOSE into an organization requires time, money, planning, and commitment!!

#*

Step 1 -Planning and Pre-Insertion Stage

- Planning - note that Software Process Planning is a KPA (Key Process Area) under Level 2 of the Capability Maturity Model (CMM)
  - Transition Plan
  - Software Development Plan
    - who, what, when, how much, risk anaylsis
- Changing existing culture
  - Understand the culture change
  - Demonstrate management commitment
    - provide time, tools, and strict enforcement
  - Sell OOSE to those who will use it



Step 2 - Insertion State

- Selecting an OO technique
  - Full life cycle, appropriate domain, target language
- Selecting a CASE tool
  - robust, adequate features
- Staffing and Organizing the project
  - Domain Analyst for reuse, OOP Guru, prototyping expert
- Training the team
  - Adequate time, dedicated time, uncompressed schedule
- Dealing with Legacy Systems
  - reverse-engineer or modify them to fit new OO

Step 3 - Project Management

- Analyze, model and prototype
  - Tendency is to over-apply OO for first project
- Effective project tracking and controlling
  - objects, classes
- Define and document the development process
- Collect software metrics
- Inspect OO software products
  - review for quality, develop good metrics, inspect quality of documentation and graphics
- Integrate the documentation

#*